CPE301 – SPRING 2019

Design Assignment 3a

Student Name: Do Le

Student #: 2001183621

Student Email: led2@unlv.nevada.edu

Primary Github address: <https://github.com/DoVietLe>

Directory: <https://github.com/DoVietLe/assignments>

The goal of the assignment is to modify the above codes to do the following:

1. Write a C AVR program that will display a string, random integer and floating-point values on the serial terminal every 1 sec. Use a FTDI chip for serial to USB conversion and display the values in the terminal.

2. Repeat 1 using a timer with interrupt for the 1 sec delay. Use a FTDI chip for serial to USB conversion and display the values in the terminal.

1. **COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS**

Components

• ATmega328PB Xplained Mini

• FTDI Serial to USB Converter

• Atmel Studio 7.0

* + Compiler
  + Debugger
  + Serial Monitor

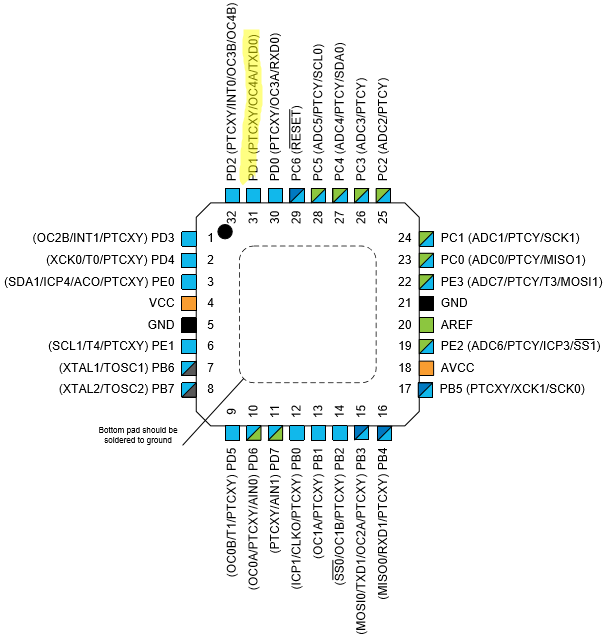
Pin Diagram

Figure 1: Pinout

PD1 used as transmit line.

1. **INITIAL/MODIFIED/DEVELOPED CODE OF TASK 1/A**

Task 1 Code

#define *F\_CPU* 16000000

#define BAUD 9600

#define UBBR\_VALUE 103

#include <avr/delay.h>

#include <avr/io.h>

void enableRandomTimer() {

// Enables timer in normal mode with a pre-scaler of 1024.

TCCR0A = 0x00;

TCCR0B = 0x05;

}

*uint8\_t* getInt() {

*uint8\_t* num = TCNT0;

num ^= (num<<(num%8));

return num;

}

float getNumber() {

float num = getInt();

num = *sqrt*(num);

return num;

}

// Enables the UART and it's transmitter.

void enableTransmit() {

// Sets port.

DDRD = 0x02;

// Sets baud rate.

UBRR0 = UBBR\_VALUE;

// Sets UART settings.

UCSR0C = 0b00000110; // to async mode with no parity

// 2 stop bits, and a frame of 8 bits.

UCSR0B = 0b00001000; // Enables transmit line.

}

// Disables the transmitter on UART.

void disableTransmit() {

UCSR0B &= ~(1<<TXEN0);

}

// Sends one byte of data out the TX line.

void sendByte(*uint8\_t* b) {

if (UCSR0B & (1<<TXEN0)) { // Checks whether transmit is enabled.

while (!(UCSR0A & (1<<UDRE0))); // Waits until the UDR0 register is ready.

UDR0 = b; // Sends the byte.

}

}

void sendString(char string[]) {

int i = 0;

while (string[i] != '\0')

sendByte(string[i++]);

}

int main(void)

{

// Declares our strings.

char myString[] = "I love CPE301";

char myIntString[10];

char myFloatString[10];

enableTransmit(); // Enables TX0.

enableRandomTimer();

while (1)

{

// Sends the string.

sendString(myString);

sendByte(' ');

// Sends the int.

*uint8\_t* myInt = getInt();

*snprintf*(myIntString, 10, "%d", myInt);

sendString(myIntString);

sendByte(' ');

// Sends the float.

float myFloat = getNumber();

*snprintf*(myFloatString, 10, "%f", myFloat);

sendString(myFloatString);

sendByte(' ');

sendByte('\r');

sendByte('\n'); // Endline.

*\_delay\_ms*(1000);

}

}

1. **DEVELOPED MODIFIED CODE OF TASK 2/A from TASK 1/A**

Task 2 Code

#define cycles\_1s 15625 // Cycles for 1s delay @16MHz with 1024 pre-scaler.

#define BAUD 9600

#define UBBR\_VALUE 103

#include <avr/interrupt.h>

#include <avr/io.h>

// Declares our strings.

volatile char myString[] = "I love CPE301";

volatile char myIntString[10];

volatile char myFloatString[10];

// Sets the interrupt.

void setInterrupt() {

// Sets timer to overflow after 1s.

OCR1A = cycles\_1s;

TCCR1A = 0b00000000; // Sets timer1 to be in CTC mode with 1024 pre-scaler.

TCCR1B = 0b00001101;

// Sets interrupts.

TIMSK1 = 0x02; // Enables interrupt on COMPA for Timer1.

sei(); // Enables global interrupt.

}

void enableRandomTimer() {

// Enables timer in normal mode with a pre-scaler of 1024.

TCCR0A = 0x00;

TCCR0B = 0x05;

}

*uint8\_t* getInt() {

*uint8\_t* num = TCNT0;

num = num ^ (num<<(num%8));

return num;

}

float getNumber() {

float num = getInt();

num = *sqrt*(num);

return num;

}

// Enables the UART and it's transmitter.

void enableTransmit() {

// Sets port.

DDRD = 0x02;

// Sets baud rate.

UBRR0 = UBBR\_VALUE;

// Sets UART settings.

UCSR0C = 0b00000110; // to async mode with no parity

// 2 stop bits, and a frame of 8 bits.

UCSR0B = 0b00001000; // Enables transmit line.

}

// Disables the transmitter on UART.

void disableTransmit() {

UCSR0B &= ~(1<<TXEN0);

}

// Sends one byte of data out the TX line.

void sendByte(*uint8\_t* b) {

if (UCSR0B & (1<<TXEN0)) { // Checks whether transmit is enabled.

while (!(UCSR0A & (1<<UDRE0))); // Waits until the UDR0 register is ready.

UDR0 = b; // Sends the byte.

}

}

void sendString(char string[]) {

int i = 0;

while (string[i] != '\0')

sendByte(string[i++]);

}

int main(void)

{

// Declares our strings.

char myString[] = "I love CPE301";

char myIntString[10];

char myFloatString[10];

enableTransmit(); // Enables TX0.

enableRandomTimer();

setInterrupt();

while (1)

{

}

}

ISR(TIMER1\_COMPA\_vect) {

// Sends the string.

sendString(myString);

sendByte(' ');

// Sends the int.

*uint8\_t* myInt = getInt();

*snprintf*(myIntString, 10, "%d", myInt);

sendString(myIntString);

sendByte(' ');

// Sends the float.

float myFloat = getNumber();

*snprintf*(myFloatString, 10, "%f", myFloat);

sendString(myFloatString);

sendByte(' ');

sendByte('\r');

sendByte('\n'); // Endline.

}

1. **SCHEMATICS**

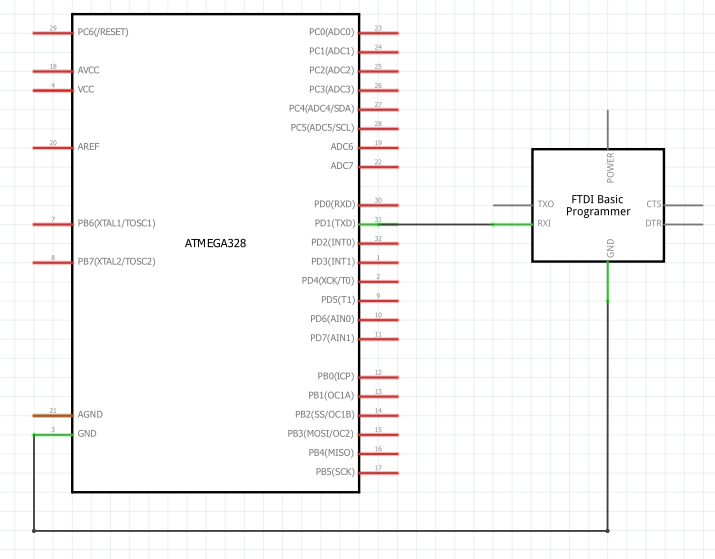


Figure 2: Breadboard and Serial to USB setup

1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**

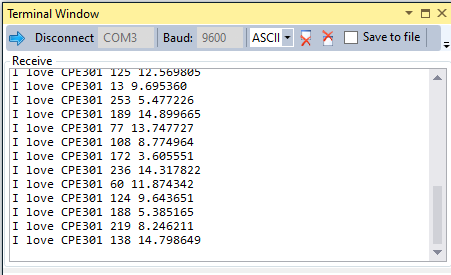


Figure 3: Task 1 output on Atmel

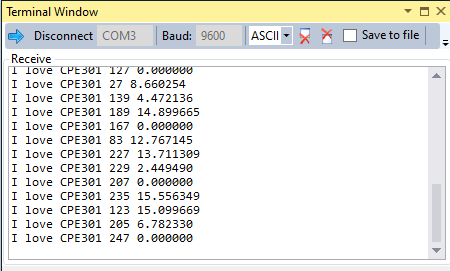


Figure 4: Task 2 output on Atmel

1. **SCREENSHOT OF EACH DEMO (BOARD SETUP)**

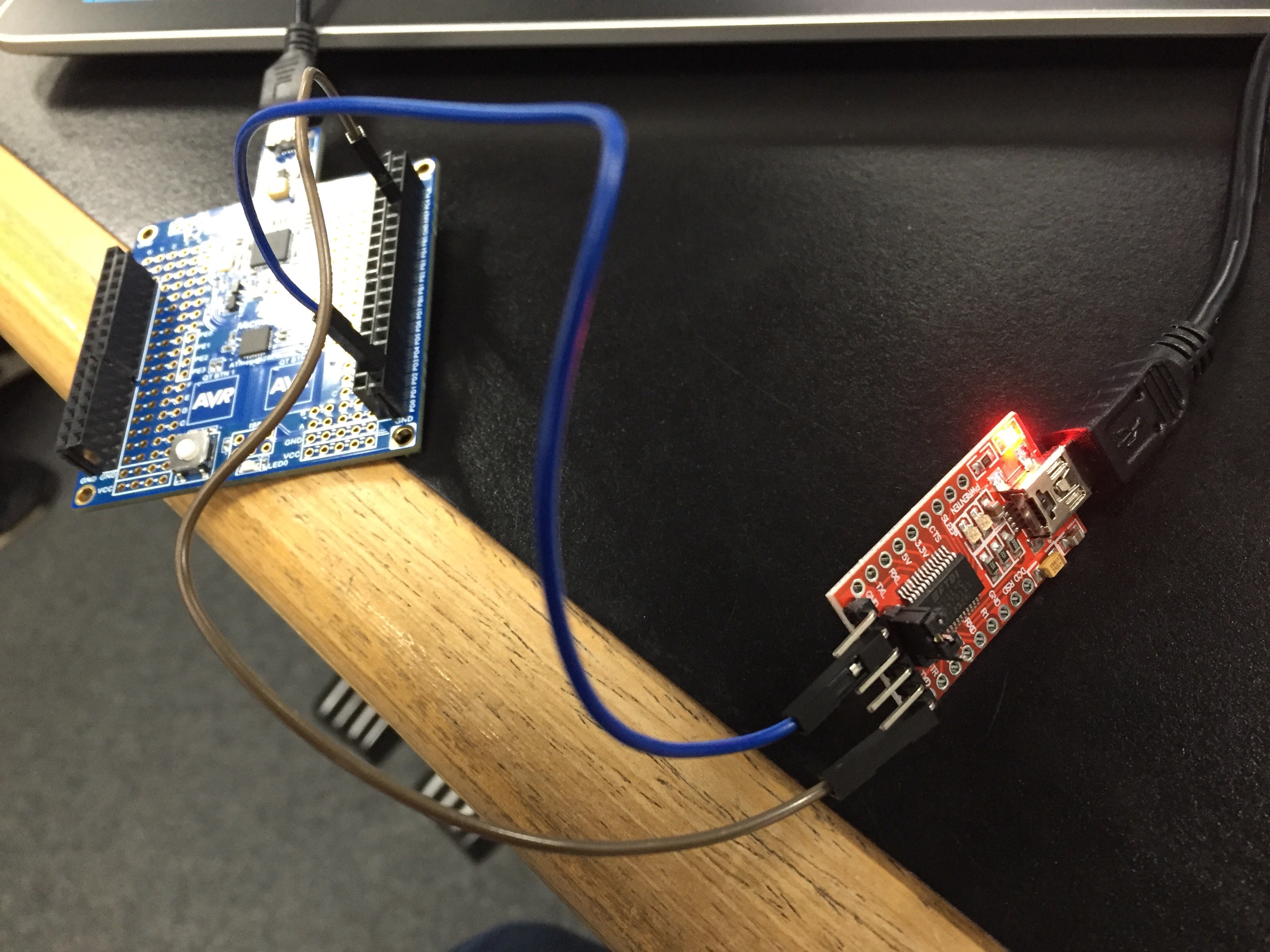


Figure 5: Board setup

1. **VIDEO LINKS OF EACH DEMO**

Task 1 Video

https://www.youtube.com/watch?v=YpImtr8pU34

Task 2 Video

https://www.youtube.com/watch?v=dnEOy225U90

1. **GITHUB LINK OF THIS DA**

https://github.com/DoVietLe/assignments/tree/master/ESD301/LAB03a

**Student Academic Misconduct Policy**

<http://studentconduct.unlv.edu/misconduct/policy.html>

“This assignment submission is my own, original work”.

Do V. Le